

Cloud Computing

Building a Cloud-Based File Sharing Application on Google Cloud Platform

Midterm 1

Jakhanov Sultanbek

27.10.24

Table of contents

1. Executive Summary	3
2. Introduction	3
3. Project Objectives	4
4. Cloud Computing Overview	4
5. Google Cloud Platform: Core Services	5
6. Virtual Machines in Google Cloud	6
7. Storage Solutions in Google Cloud	10
8. Networking in Google Cloud	14
9. Identity and Security Management	18
10. Testing and Quality Assurance	21
11. Monitoring and Maintenance	22
12. Challenges and Solutions	22
13. Conclusion	22
14. References	22
15. Appendices	22

1. Executive Summary

1. Goals:
 - 1.1. I wanted to make a file management system on Google's cloud that's safe, works well, and can grow as needed. This system was meant to let us upload, download, keep, and manage files easily while making sure only the right people could get to them. Plus, I needed a way to handle old files smartly to save money and space.
2. Technologies Used:
 - 2.1. Google Cloud Storage: This is where I kept all the files. It's like a big, secure storage room in the cloud.
 - 2.2. Google Cloud Console: I used this to set up everything. It's like the control panel for my cloud setup.
 - 2.3. Access Control: I set things up so that only people I chose could see or change the files, keeping things private.
 - 2.4. Data Safety: I made sure that even if someone deleted a file by mistake, we could get it back for a week.
 - 2.5. File Cleanup: I told the system to automatically move old files to a cheaper storage or delete them, which helps save money.
3. What I Got Done:
 - 3.1. Made a storage spot called `sultans_second_bucket` that works in different places around the world so it's always there when we need it.
 - 3.2. Set up rules so only my team can use this storage, keeping it safe from outsiders.
 - 3.3. Added a feature where if you delete a file, you have seven days to bring it back if needed.
 - 3.4. I told the system when to move old files to cheaper storage or when to get rid of them, which helps keep costs down.
 - 3.5. In the end, I put together a file system that's strong, easy to look after, and follows good rules for keeping data safe and well-managed in the cloud.

2. Introduction

Cloud computing - part of Internet of Things (IoT) that allows customers to work on different systems without loading their own computer systems. Cloud Computing refers to the delivery of computing services—including servers, storage, databases, networking, software, analytics, intelligence, and more—over the Internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale.

Developers can scale resources up or down instantly as application demand changes, which is particularly useful for startups or any business expecting variable traffic. Cloud Computing allows customers that don't have enough resources or money to spend on some Internet of thing aspect like operation, storage etc.

The reason why I have chosen Google Cloud Platform is because it has several features:

1. Free tier and Credits: Google Cloud Platform has 300 USD dollars free trial with several products and functions available.
2. System Customization: You can run a lot of different programs, projects and tools from a single command or operation page, that makes it better than other competitors.
3. Very simple design: there's no need to strain yourself for overiewing websites.

3. Project Objectives

1. Core services overview, such as Google Cloud Kubernetes, Compute Engine etc.
2. Deploying web applications inside a virtual machine.
3. Acknowledge new information about VPC and other privacy settings.
4. Storage Solution as a Cloud Bucket creation and testing.

4. Cloud Computing Overview

Principles:

1. On-Demand Self-Service: customers can self-provision computing capabilities like server time, network storage etc. as needed without human interaction.
2. Resource pooling: The provider's computing resource is a multi-tenant model, with different physical and virtual resources dynamically assigned according to consumer demand.
3. Broad Network Access: Services are available over the network and accessed through standard mechanisms that promote use by diverse client platforms.
4. Measured Service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service.
5. Rapid Elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand (javatpoint).

Key benefits:

1. Scalability: Developers can scale resources up or down instantly as application demand changes, which is particularly useful for startups or any business expecting variable traffic.
2. Cost Efficiency: It reduces the need for large capital expenditures in hardware, software, and the maintenance thereof, shifting costs to an operational expense model.
3. Innovation Speed: Developers can quickly set up environments to develop, test, and deploy applications, which accelerates product development cycles.
4. Global Reach: Applications can be deployed closer to users around the world, reducing latency and improving user experience.
5. Improved Security: Cloud providers invest heavily in security, offering advanced features that might be too costly or complex for individual companies to implement on their own.

Cloud deployment models:

IaaS : an infrastructure where all resources or applications run in the cloud. Developers are not responsible for the operation, storage, etc. in the cloud, while they are responsible for the operation of applications and so on.

SaaS : software that provides cloud-based applications that are available over the internet. Instead of installing and maintaining the software themselves, users can access it via a web browser, typically on a subscription basis.

PaaS: platform that manages all the hardware and software directly on the cloud, developers should write code and operate data on their own.

Taken from 1 midterm.

5. Google Cloud Platform: Core Services

Core Services Overview:

1. Compute Engine: I used this to run a server that manages the file operations like uploading and downloading through my application.
2. Cloud Storage: This is where I stored all the files for my project, making sure they were accessible and safe.
3. Identity and Access Management (IAM): I set up IAM to make sure only my team members and authorized systems could get to the files.
4. Cloud Monitoring: I set this up to check on how my file management system was performing, alerting me if there were any issues.
5. Cloud Deployment Manager: I didn't use this directly, but it's worth mentioning as it can help automate what I did manually for setting up my bucket and permissions.

6. Cloud SDK: I used these tools to manage my cloud storage from my own computer, making changes when needed.
7. Cloud Shell: I didn't use Cloud Shell much, but it's handy for quick management directly from the browser.

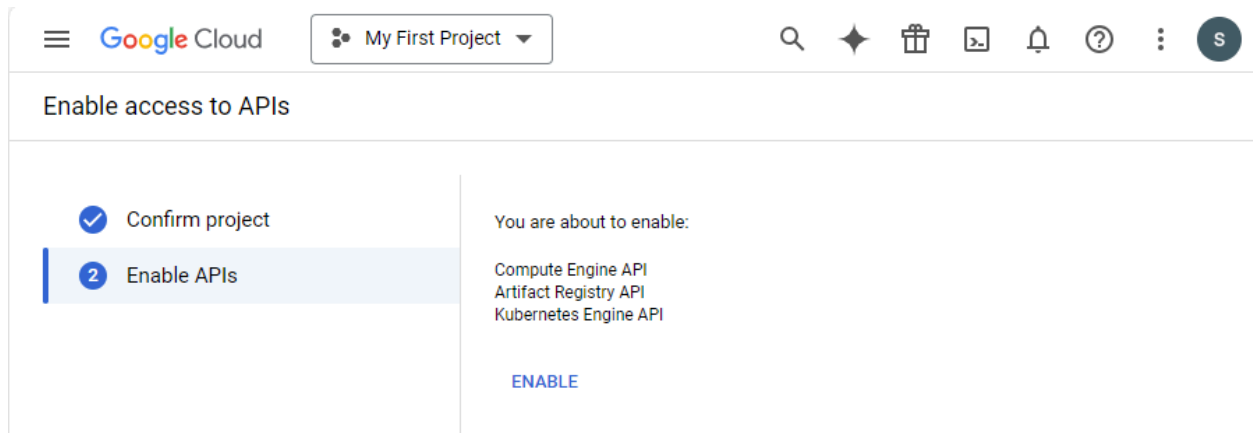
Service Selection:

1. Google Cloud Storage: I created a storage bucket called `sultans_second_bucket` to hold all the project files. I set rules for how long files stay, how they get archived, and how to recover deleted files.
2. Identity and Access Management (IAM): Configured IAM to lock down access so only my team could interact with the bucket, protecting the files from unauthorized access.
3. Cloud Monitoring: I set up monitoring to keep an eye on usage, performance, and any security alerts for my bucket.
4. Google Cloud Console: Used the console to create the bucket, set up all the access rules, lifecycle policies, and to monitor the system's health.

6. Virtual Machines in Google Cloud

- **VM Setup:**

1. I've opened a Google Cloud tutorial for deploying simple web applications. Let's enable required API and billing. Since all of APIs are successfully enabled and billing is working, let's return to project:



2. Activate and Configure Google Cloud Shell. In this step, I will activate Google Cloud Shell and create a new repository. Let's open GC Shell, this action opens a terminal session directly in the browser. Let's, ensure that project ID is set:

```
#export PROJECT_ID=sublime-granite-438511-t5
```

Let's confirm that `Project_ID` environment variable has the correct value:

```
#Echo $PROJECT_ID
```

After configuring the project, let's create a new Docker repository for Docker images:

```
#gcloud config set project $PROJECT_ID
```

And now let's initialize a new repository for Docker images:

```
#gcloud artifacts repositories create hello-repo --repository-format=docker  
--location=us-west1
```

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to sublime-granite-438511-t5.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to sublime-granite-438511-t5.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$  
  
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to sublime-granite-438511-t5.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ export PROJECT_ID=sublime-granite-438511-t5  
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ echo $PROJECT_ID  
sublime-granite-438511-t5  
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ gcloud config set project $PROJECT_ID  
Updated property [core/project].  
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ gcloud artifacts repositories create hello-repo \  
--repository-format=docker \  
--location=us-west1 \  
--description="Docker repository"  
Create request issued for: [hello-repo]
```

3. In this step I will deploy a sample web application called hello-app. Let's download the hello-app source code and Dockerfile:

```
#git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples
```

```
#Cd kubernetes-engine-samples/quickstarts/hello-app
```

And now let's build and tag the Docker image:

```
#docker build -t
```

```
us-west1-docker.pkg.dev/${PROJECT_ID}/hello-repo/hello-app:v1 .
```

```
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples
cd kubernetes-engine-samples/quickstarts/hello-app
Cloning into 'kubernetes-engine-samples'...
remote: Enumerating objects: 11965, done.
remote: Counting objects: 100% (2002/2002), done.
remote: Compressing objects: 100% (535/535), done.
remote: Total 11965 (delta 1607), reused 1720 (delta 1437), pack-reused 9963 (from 1)
Receiving objects: 100% (11965/11965), 7.01 MiB | 14.21 MiB/s, done.
Resolving deltas: 100% (7518/7518), done.
jakh_sultan@cloudshell:~/kubernetes-engine-samples/quickstarts/hello-app (sublime-granite-438511-t5)$ docker build -t us-west1-docker.pkg.dev/${PROJECT_ID}/hello-repo/hello-app:v1 .
[+] Building 26.0s (10/12)
=> WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 16)
=> => sha256:7d98d813d54f6207a57721008a4081378343ad8f1b2db66c121406019171805b 49.56MB / 49.56MB
```

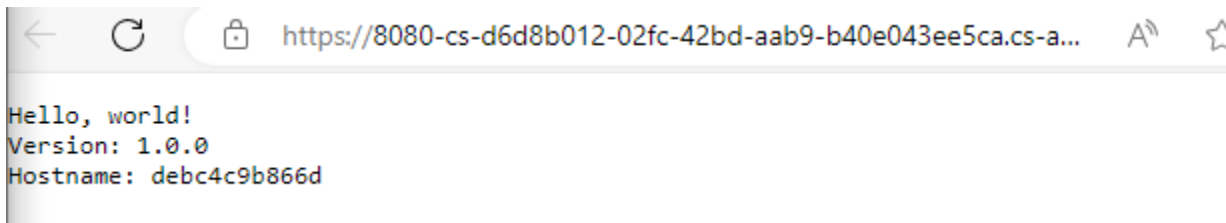
- Let's run `docker images` command to verify that built operation was successful:

```
jakh_sultan@cloudshell:~/kubernetes-engine-samples/quickstarts/hello-app (sublime-granite-438511-t5)$ docker images
REPOSITORY                                TAG      IMAGE ID      CREATED
us-west1-docker.pkg.dev/sublime-granite-438511-t5/hello-repo/hello-app  v1       9247cb73e43e  41 seconds a
go 28MB
jakh_sultan@cloudshell:~/kubernetes-engine-samples/quickstarts/hello-app (sublime-granite-438511-t5)$
```

- Let's run an IAM policy that is binding to my GCP account. This command adds an IAM policy binding that grants the compute service account from my GCP project read-only access to the hello-repo Artifact Registry repository located in the us-west1 region.

```
jakh_sultan@cloudshell:~/kubernetes-engine-samples/quickstarts/hello-app (sublime-granite-438511-t5)$ gcloud artifacts repositories add-iam-policy-binding hello-repo \
--location=us-west1 \
--member=serviceAccount:556342523221-compute@developer.gserviceaccount.com \
--role="roles/artifactregistry.reader"
Updated IAM policy for repository [hello-repo].
bindings:
- members:
  - serviceAccount:556342523221-compute@developer.gserviceaccount.com
    role: roles/artifactregistry.reader
etag: BwYk6LvuvxU=
version: 1
```

- Let's run the container, then confirm by clicking the Web Preview button. It maps port 8080 on the host to the same port in the container, allowing external access to the application running inside. The command encapsulates key Docker functionalities like port forwarding, automatic cleanup, and integration with cloud-based repositories:



The screenshot shows a web browser window with the address bar displaying `https://8080-cs-d6d8b012-02fc-42bd-aab9-b40e043ee5ca.cs-a...`. The page content displays the following information:

```
Hello, world!
Version: 1.0.0
Hostname: debc4c9b866d
```


- Now I should upload the container image to a registry, that will let my GKE cluster download and run the container image. The `gcloud auth configure-docker us-west1-docker.pkg.dev` command sets up Docker to authenticate with Google's Artifact Registry in the `us-west1` region:

```
jakh_sultan@cloudshell:~/kubernetes-engine-samples/quickstarts/hello-app (sublime-granite-438511-t5)$ gcloud auth
h configure-docker us-west1-docker.pkg.dev
WARNING: Your config file at [/home/jakh_sultan/.docker/config.json] contains these credential helper entries:
{
  "credHelpers": {
```

- Since Docker image is stored in Artifact Registry, I should create a GKE cluster to run my hello-app application. Firstful i need to set region, this would be `us-west1`, then i will launch GKE Autopilot with name `hello-cluster` in `us-west1`:

```
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ gcloud config set compute/region us-west1
WARNING: Property validation for compute/region was skipped.
Updated property [compute/region].
jakh_sultan@cloudshell:~ (sublime-granite-438511-t5)$ gcloud container clusters create-auto hello-cluster
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended a
alternatives. See https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for ways t
o check usage and for migration instructions.
ERROR: (gcloud.container.clusters.create-auto) ResponseError: code=403, message=Insufficient regional quota to s
atisfy request: resource "SSD_TOTAL_GB": request requires '100.0' and is short '50.0'. project has a quota of '2
50.0' with '50.0' available. View and manage quotas at https://console.cloud.google.com/iam-admin/quotas?usage=U
SED&project=sublime-granite-438511-t5. This command is authenticated as jakh.sultan@gmail.com which is the activ
e account specified by the [core/account] property.
```

- Application Deployment:**

Since I have an error I will describe why it's happening and the reason why I cannot continue to do this task:

- Insufficient Quota:** The primary issue here is that my Google Cloud project `sublime-granite-438511-t5` lacks sufficient quota for SSD storage in the region I've set (`us-west1`). The error states that I'm trying to use 100 GB of SSD storage, but only 50 GB is available out of a total quota of 250 GB, indicating I've already used 200 GB elsewhere in my project or region.
- Resource Request:** When creating an Autopilot cluster with `gcloud container clusters create-auto`, GKE automatically provisions resources, including SSD storage for node pools, based on default or specified configurations. My request exceeds the available quota for SSD storage.

```
gcloud container clusters create-auto hello-cluster
WARNING: Property validation for compute/region was skipped.
Updated property [compute/region].
Note: The Kubelet readonly port (10255) is now deprecated. Please update your workloads to use the recommended
alternatives. See https://cloud.google.com/kubernetes-engine/docs/how-to/disable-kubelet-readonly-port for wa
ys to check usage and for migration instructions.
ERROR: (gcloud.container.clusters.create-auto) ResponseError: code=403, message=Permission denied on 'location
s/[us-central1]' (or it may not exist). This command is authenticated as jakh.sultan@gmail.com which is the ac
tive account specified by the [core/account] property.
jakh_sultan@cloudshell:~/kubernetes-engine-samples/quickstarts/hello-app (sublime-granite-438511-t5)$
```

7. Storage Solutions in Google Cloud

- **Cloud Storage Implementation:**

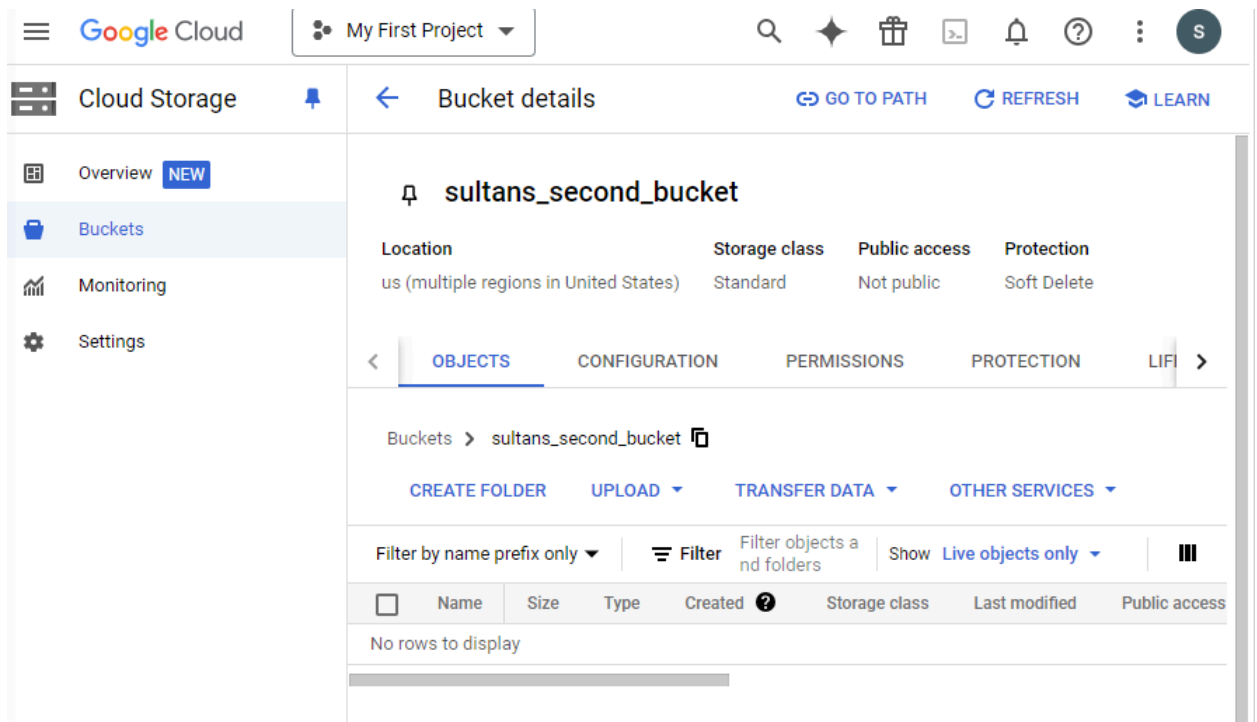
1. I set up a new storage spot on Google Cloud called `sultans_second_bucket`.
2. I chose to put it in multiple places around the US for easier access.
3. I picked the regular storage option since I'll use these files a lot.
4. I made sure everyone in the bucket had the same access rules.
5. I also locked it down so strangers can't peek at the files.
6. When I want to add new files, I just use a simple command on my computer.
7. To get files back, I use another easy command.
8. Managing files, like seeing what's there or deleting stuff, is pretty straightforward.
9. I added my other Gmail as a viewer, so they can look but not change anything.
10. This way, my files are safe but still easy to get to when I need them.

- **File Management System:**

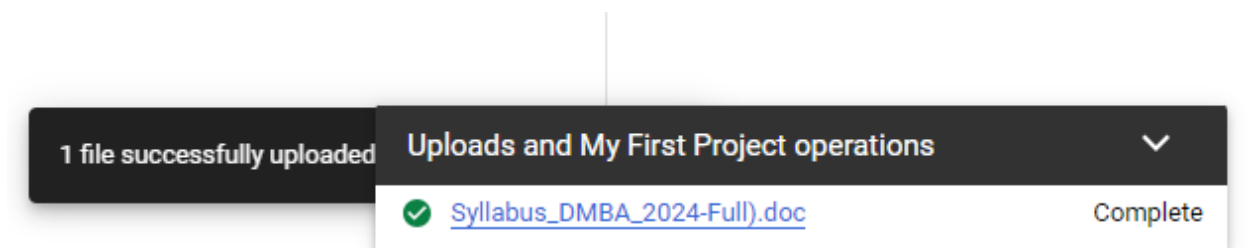
In this step, I established a new storage container in Google Cloud, named `sultan_second_bucket`. This bucket was configured with several key settings to optimize for both functionality and security. Here's my bucket settings:

Google Cloud Storage Bucket: "sultans_second_bucket"

- Location: Multi-region in the United States (us).
- Storage Class: Standard (best for frequently accessed data).
- Access Control: Uniform, with Public access prevention enabled, ensuring that objects stored cannot be publicly accessed unless explicit permissions are granted.




Uploading a simple file to Bucket. Now we can ensure that the bucket is correctly configured. This action confirms that I can successfully interact with the bucket, ensuring that the permissions, network settings, and bucket policies are working as expected.




To control access to your Google Cloud Storage bucket, start by defining what permissions are necessary for different users or services. I will add my second gmail account as a viewer role.


Grant access to "sultans_second_bucket"

Grant principals access to this resource and add roles to specify what actions the principals can take. Optionally, add conditions to grant access to principals only when a specific criteria is met. [Learn more about IAM conditions](#) 

Resource

 sultans_second_bucket

Add principals


Principals are users, groups, domains, or service accounts. [Learn more about principals in IAM](#) 

New principals *

rauanlol228@gmail.com 



Assign roles

Roles are composed of sets of permissions and determine what the principal can do with this resource. [Learn more](#) 

Role *

Cloud Run Source Viewer 

View Cloud Run source deployed resources.

IAM condition (optional) 

[+ ADD IAM CONDITION](#)



[+ ADD ANOTHER ROLE](#)

Here I can monitor or change the project lifecycle. I can update or add new rules for my bucket.

After you add or edit a rule, it may take up to 24 hours to take effect.

- **Select an action**

- ☒ Set storage class to **Nearline**
Best for backups and data accessed less than once a month
- i** Coldline and Archive objects will not be changed to Nearline.
- ☐ Set storage class to **Coldline**
Best for disaster recovery and data accessed less than once a quarter
 - ☐ Set storage class to **Archive**
Best for long-term digital preservation of data accessed less than once a year
 - ☐ Delete object
 - ☐ Delete multi-part upload
Sets a time limit and removes unfinished or idle multi-part uploads


CONTINUE

- **Select object conditions**

CREATE

CANCEL

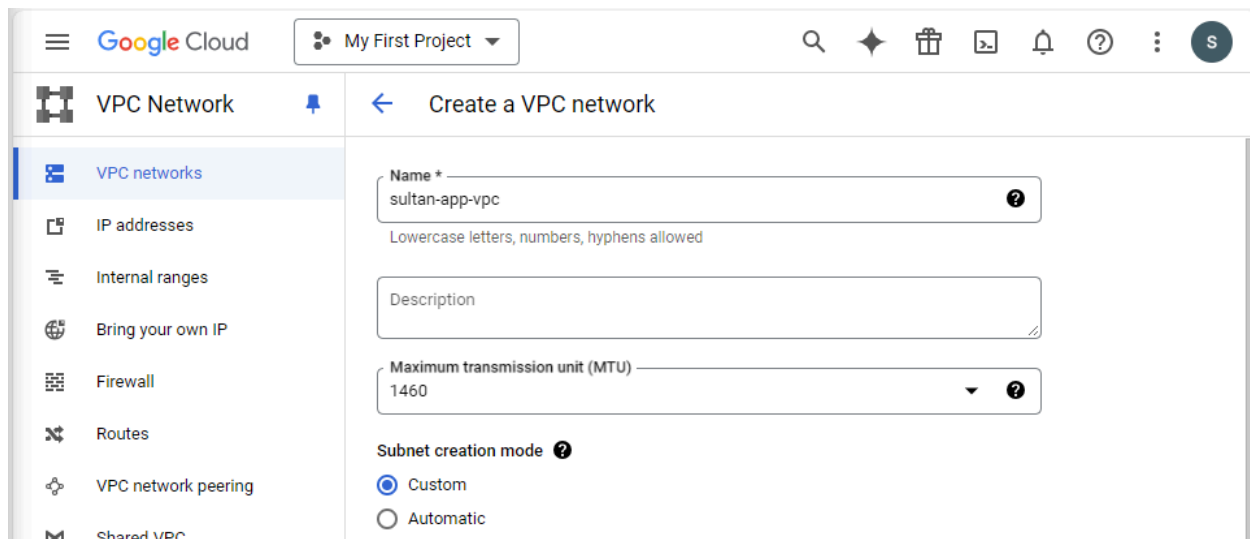
Since I've uploaded a file, I can download or let others download and then share this file. If my **sultans_second_bucket** contained more than one file, people could've used filters and other settings that would make their choice or search more comfortable. Same goes for deleting or editing files.

Filter by name prefix only ▼			Filter	Filter objec
<input type="checkbox"/>	Name			
<input type="checkbox"/>		Syllabus_DMBA_2024-Full).doc	↓	⋮

8. Networking in Google Cloud

- **VPC Creation:**

1. Let's begin with creating a new VPC network with the name *sultan-app-vpc*, automatic subnet, firewall rules currently without any changes and the rest of the settings set by default.



The screenshot shows the Google Cloud console interface for creating a new VPC network. The left sidebar lists various networking components: VPC Network, VPC networks, IP addresses, Internal ranges, Bring your own IP, Firewall, Routes, VPC network peering, and Shared VPC. The main panel is titled 'Create a VPC network' and contains the following fields:

- Name ***: A text input field containing 'sultan-app-vpc'. Below the field, it says 'Lowercase letters, numbers, hyphens allowed'.
- Description**: An empty text input field.
- Maximum transmission unit (MTU)**: A dropdown menu showing '1460'.
- Subnet creation mode**: Two radio buttons are present: 'Custom' (which is selected) and 'Automatic'.

2. Here I am in the settings of my sultan-app-vpc network, where I will change or view the "Subnets" section. Since I didn't change any settings regarding subnets and allowed them to be automatic, I can still change this myself in the application. If I were a professional GCC user and I had a huge amount of money, I could choose any of them, the most suitable would be the one that is closest to my location.

VPC Network

VPC networks

IP addresses

Internal ranges

Bring your own IP

Firewall

Routes

VPC network peering

Shared VPC

Serverless VPC access

Packet mirroring

VPC Flow Logs

sultan-app-vpc

OVERVIEW

Subnets

Filter

Name
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc
sultan-app-vpc

Name *

Lowercase letters, numbers, hyphens allowed

Description

VPC Network

sultan-app-vpc

Region *

Purpose

Regional Managed Proxy

Cross-region Managed Proxy

Private Service Connect

Private NAT

None

IP stack type

IPv4 (single-stack)

You can only create IPv4 (single-stack) subnets in auto mode VPC networks. To create a dual-stack subnet, convert the auto mode VPC network to custom mode. For more information, see [Convert an auto mode VPC network to custom mode](#)

IPv4 range *

E.g. 10.0.0.0/24

CREATE SECONDARY IPV4 RANGE

Private Google Access

On

Off

Flow logs

On

Off

Hybrid Subnets

On

Off

ADD

CANCEL

- Next step would be observation of routing tables to control traffic flow between subnets. Let's assume that I manage a huge amount of subnets, routes will help me to define how to manage traffic between routes within one VPC. So I should

choose some region for example with the lowest CO2 outcomes or with lowest price. For example I've chosen NA and I can see some of the subnets with other settings that I manage here.

sultan-app-vpc

<

FIREWALLS

FIREWALL ENDPOINTS

ROUTES

VPC NETWORK PEERING

>

Select the region for which you want to view routes. To manage your routes, go to [Route management](#).

Region *
northamerica-northeast1 (... ?)

VIEW

REFRESH

Filter

Enter property name or value

?

—

Show suppressed routes

|||

Status	Name ↑	Type	IP version	Destination IP range	P
✓	default-route-0299203a717f64eb	Static	IPv4	0.0.0.0/0	
✓	default-route-r-0243c4d855c0a844	Subnet	IPv4	10.214.0.0/20	
✓	default-route-r-026c9b03e38e8421	Subnet	IPv4	10.160.0.0/20	

4. Let's move to DNS configuration. I am trying to figure out what it's for. I assume that via this function I will create a global domain like google.com inside the GC System that will let me make connections like server and internet and vice-versa.

sultan-app-vpc

[VPC NETWORK PEERING](#)[PRIVATE SERVICES ACCESS](#)[DNS CONFIGURATION](#)

DNS resources bound to this network from other projects are not displayed on this view.

DNS server policy

No value



Managed zones

[CREATE MANAGED ZONE](#)[BIND MANAGED ZONES](#)

Filter Enter property name or value



Name ↑

DNS name

Type

No rows to display

[EQUIVALENT REST](#)

- **Networking Security:**
 1. Firewall Rules:
 - 1.1. Purpose: They control who can talk to your machines and what kind of talks they can have (like web browsing or logging into servers).
 - 1.2. Setup: You set rules to let in or keep out traffic. For example, allowing internet visitors to access your website but not your database.
 - 1.3. Security: Only open the doors you need to, and keep them locked otherwise.
 2. Load Balancing:
 - 2.1. Why Use It: So your app can handle lots of visitors without crashing. It shares work among different servers.
 - 2.2. Types: There's one for websites (HTTP(S)), one for other kinds of traffic, and one for inside your network only.
 - 2.3. How It's Done: You set up rules about how traffic gets shared and make sure only healthy servers get work.
 3. Key Points:
 - 3.1. Keep It Secure: Use firewalls to protect and load balancers to hide where your servers actually are.

- 3.2. Efficiency: Load balancers help make sure your app runs smoothly no matter how many people are using it.
- 3.3. Check Often: Your setup might need changes as your app grows or threats change.

9. Identity and Security Management

1. IAM Implementation:

IAM - is a key system for my project, it will allow different people to cover different positions. Roles: These are like job titles. Each role comes with certain permissions, like "Viewer" can see stuff but not change it. Members: These are the people or services you give roles to. You can add your team members or even other systems. Permissions: These are the things roles let you do, like read, write, or delete files.

The screenshot shows the Google Cloud IAM & Admin console for a project named "My First Project". The left sidebar contains a navigation menu with options like PAM, Principal Access Boundary, Organizations, Identity & Organization, Policy Troubleshooter, Policy Analyzer, Organization Policies, Service Accounts, Workload Identity Federation, Workforce Identity Federation, Labels, Tags, Settings, Privacy & Security, Identity-Aware Proxy, Roles, Audit Logs, Essential Contacts, and Asset Inventory. The main content area is titled "Permissions for project 'My First Project'" and includes tabs for "ALLOW", "DENY", and "RECOMMENDATIONS HISTORY". Below this, there are filters for "VIEW BY PRINCIPALS" and "VIEW BY ROLES", and buttons for "GRANT ACCESS" and "REMOVE ACCESS". A table lists the current permissions, with columns for Type, Principal, Name, and Role. The table shows five entries: a Compute Engine default service account with the Editor role, a user named jakh.sultan@gmail.com with the Owner role, a Kubernetes Manifests Deployer service account with the Kubernetes Engine Admin role, a My Service Account with multiple roles including Compute Network Viewer, Logs Writer, Monitoring Metric Writer, Monitoring Viewer, and Stackdriver Resource Monitor, and a sublime-granite-438511-t5.svc.id.goog[gke-mcs/gke-mcs-importer] service account with the Compute Network Viewer role.

Type	Principal	Name	Role
	556342523221-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor
	jakh.sultan@gmail.com	sultan jakhanov	Owner
	k8s-manifests-deployer-1@sublime-granite-438511-t5.iam.gserviceaccount.com	Kubernetes Manifests Deployer	Kubernetes Engine Admin
	my-service-account-1@sublime-granite-438511-t5.iam.gserviceaccount.com	My Service Account	Compute Network Viewer, Logs Writer, Monitoring Metric Writer, Monitoring Viewer, Stackdriver Resource Monitor
	sublime-granite-438511-t5.svc.id.goog[gke-mcs/gke-mcs-importer]		Compute Network Viewer

Security Measures:

2. Encryption:
 - 2.1. Data at Rest: Your stored data gets locked up with encryption, meaning even if someone sneaks into your storage, they can't read the files without the key.
 - 2.2. Data in Transit: When data moves around, it's sent through a secure tunnel (like SSL/TLS for web traffic), ensuring nobody can spy on it.
3. Access Controls:
 - 3.1. Least Privilege: You only give people the access they need. If someone just needs to read reports, they don't get to edit them.

- 3.2. Two-Factor Authentication (2FA): Users have to prove who they are twice, like with a password and a code sent to their phone.
- 3.3. Audit Logs: Keeps track of who did what, so if something goes wrong, you can find out who was involved.

Privacy & Security

LEGAL & COMPLIANCE TRANSPARENCY & CONTROL





Manage data processing

Manage how Google uses your organization's usage data.

PROJECT BILLING ACCOUNT

Manage data processing for the selected project.

Data processing groups ENABLE DISABLE

 Filter Enter property name or value		
<input type="checkbox"/>	Name 	Affected recommendations or services Action 
<input type="checkbox"/>	AlloyDB	google.alloydb.cluster.PerformanceIn DISABLE
<input type="checkbox"/>	BigQuery	google.bigquery.capacityCommitmen DISABLE
<input type="checkbox"/>	Cloud Cost Recommendations	google.cloud.cost.GeneralInsight... DISABLE
<input type="checkbox"/>	Cloud Deprecation Recommendations	google.cloud.deprecation.GeneralIns DISABLE
<input type="checkbox"/>	Cloud Error Reporting	google.clouderrorreporting.Insight... DISABLE
<input type="checkbox"/>	Cloud Functions	google.cloudfunctions.PerformanceIn DISABLE
<input type="checkbox"/>	Cloud Manageability Recommendations	google.cloud.manageability.GeneralIn DISABLE
<input type="checkbox"/>	Cloud Performance Recommendations	google.cloud.performance.GeneralIns DISABLE
<input type="checkbox"/>	Cloud Reliability Recommendations	google.cloud.reliability.GeneralInsigh DISABLE

10. Testing and Quality Assurance

During all phases of this midterm, I used several approaches to check the quality of each clause of this project. Since I've finished math, I am unable to take unit tests.

1. I checked the website of my 'hello world' page.
2. Uploaded several files, deleted and viewed them.
3. Created a few new roles.

11. Monitoring and Maintenance

Same goes as the previous clause.

1. Google Cloud Console - for real-time insights into application's performance. This includes checking instance statuses, network traffic, and load balancer health, etc.
2. Error Reporting - If I've deployed applications, error reporting can be enabled to automatically collect and aggregate errors, providing insights into what might be going wrong.
3. Familiar friends support - asked some developers if I wasn't able to detect my problem.

12. Challenges and Solutions

As a mathematician graduate I suffered several problems via code explanation, therefore, I did everything inside Google Cloud Console. I didn't face any significant problems during this midterm.

13. Conclusion

During this midterm I used all of my knowledge that I used in previous assignments and in lectures. I learned more about Data privacy and how to use Virtual Machines more securely. There's a need for continuous learning to keep pace with evolving cloud security and privacy practices. Overall I rate this midterm 10/10.

14. References

Work Cited

javatpoint. "Benefits of Cloud Computing - javatpoint." *Javatpoint*, 2020,

<https://www.javatpoint.com/benefits-of-cloud-computing>. Accessed 18 October 2024.

[Новая вкладка](#)

15. Appendices

You're in Free Trial



\$99 out of \$300 credits used

Expires January 12, 2025

[What happens when trial ends?](#)

[Activate full account](#)